## Amendments to the Specification:

Please insert the following paragraphs before the first paragraph on page 5:

An aspect of the present invention is based on generating differential log records using the bit-wise exclusive-OR (XOR) operations, and applying the differential log records in an arbitrary order to roll forward for the redo operation and roll back for the undo operation. That the execution order is independent from the order of log record creation is based on the following mathematical observations.

## Definition of the differential log record

If an update operation using transaction changes the value  $b_{t-1}$  of a slot (the slot denotes a part of the database; it can be a field, a record, or any other database object) to  $b_t$ , then the corresponding differential log record  $\Delta_t$  is defined as  $b_t \oplus b_{t-1}$ , where  $b_t$  is the slot after the t-th update occurs,  $\oplus$  denotes the bit-wise XOR operation, and  $b_{t-1}$  is the slot before the t-th update occurs.

# Theorem 1: Recoverability (for redo and undo) using differential log records

Assume that the value of a slot has changed from  $b_0$  to  $b_m$  by m number of updates, and each update  $u_t$  (t = 1, ..., m) has generated the differential log record  $\Delta_t$ . Then, the final

Docket No. K-0254

value b<sub>m</sub> can be recovered (for the redo operation) from using the initial value b<sub>0</sub> and the differential log records. In the same way, the initial value b<sub>0</sub> can be recovered (for the undo operation) from using the final value b<sub>m</sub> and the differential log records.

### (Proof)

XOR operations are commutative and associative. For any binary numbers, p, q, and r,

#### Therefore,

i) Recovery of b<sub>m</sub> from b<sub>0</sub>:

 $\underline{b_0} \oplus \underline{\Delta_1} \oplus \underline{\Delta_2} \oplus ... \oplus \underline{\Delta_m}$ 

 $=b_0\oplus (b_0\oplus b_1)\oplus (b_1\oplus b_2)\oplus ...\oplus (b_{m-1}\oplus b_m)$ 

 $= (b_0 \oplus b_0) \oplus (b_1 \oplus b_1) \oplus ... \oplus b_m$ 

 $= 0 \oplus ... \oplus 0 \oplus b_m$ 

 $= b_m$ 

ii) Recovery of b<sub>0</sub> from b<sub>m</sub>:

 $\underline{b_m \oplus \Delta_m \oplus \Delta_{m-1} \oplus ... \oplus \Delta_1}$ 

 $= b_m \oplus (b_{m-1} \oplus b_m) \oplus ... \oplus (b_0 \oplus b_1)$ 

 $= b_m \oplus (b_m \oplus b_{m-1}) \oplus ... \oplus (b_1 \oplus b_0)$ 

 $= (b_m \oplus b_m) \oplus (b_{m-1} \oplus b_{m-1}) \oplus ... \oplus b_0)$ 

 $\underline{= 0 \oplus ... \oplus 0 \oplus b_0}$ 

 $= b_0$ 

# Theorem 2: Order-independence of redo and undo operations

For redo operations, given the initial value  $b_0$  of a slot and differential log records  $\Delta_t$ , where  $t=1,\ldots,m$ , the final value  $b_m$  can be recovered (for the redo operation) from applying the differential log records in an arbitrary order. In the same way, the initial value  $b_0$  can be recovered (for the undo operation) from applying the differential log records in an arbitrary order.

### (Proof)

Assume that differential log records are applied in the order of  $\Delta_{(1)}$ ,  $\Delta_{(2)}$ , ...,  $\Delta_{(m)}$  where  $\Delta_{(t)}$  is selected in an arbitrary order from the set of sequentially-generated differential log records  $\{\Delta_1, \ldots, \Delta_{m-1}, \Delta_m\}$ .

Then, the final value of the slot is

$$\underline{b_0 \oplus \Delta_{(1)} \oplus \Delta_{(2)} \oplus ... \oplus \Delta_{(m)}}$$

Since  $\oplus$  is commutative the order of applying  $\Delta_{(k)}$  can be changed into the following sequence.

$$b_0 \oplus \Delta_1 \oplus \Delta_2 \oplus ... \oplus \Delta_m$$

This completes the proof for the redo operation.

The same proof applies to the undo operations.

# Theorem 3: Order-independence between redo and undo operations

Assume that there are n differential log records to be redone, whose redo operations are denoted by  $R_i$  (i = 1, ..., n), and m differential log records to be undone, whose undo operations are denoted by  $U_i$  (j = 1, ..., m). Then, all the following execution sequences result in the same state of the database.

Sequence 1. (undo phase after redo phase) R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub>, U<sub>1</sub>, U<sub>2</sub>, ..., U<sub>m</sub>

Sequence 2. (redo phase after undo phase) U<sub>1</sub>, U<sub>2</sub>, ..., U<sub>m</sub>, R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub>

Sequence 3. (redo and undo in a single phase) any permutation of { R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub>, U<sub>1</sub>, U<sub>2</sub>, ..., U<sub>m</sub> } (for example, R<sub>1</sub>, U<sub>1</sub>, R<sub>2</sub>, R<sub>3</sub>, U<sub>2</sub>, ..., R<sub>n</sub>, ..., U<sub>m</sub>)

### (Proof)

The theorem results from the commutative and associative properties of XOR operations involved in the redo and undo operations.